REMARKS

I. Preliminary Matter

Applicants thank the Examiner for the courtesy and professionalism extended during the telephonic interview with Applicants' representative on September 19, 2007. As a preliminary matter, Applicants believe that the Office Action mailed August 1, 2007 ("Office Action") is incomplete with respect to the § 102(b) and § 103(a) rejections because the Examiner repeated the rejections without having considered Applicants' arguments. As M.P.E.P. § 707.07(f) makes clear:

> In order to provide a complete application file history and to enhance the clarity of the prosecution history record, an examiner must provide clear explanations of all actions taken by the examiner during prosecution of an application.
>
> . . .
>
> Where the applicant traverses any rejection, the examiner should, if he or she repeats the rejection, take note of the applicant's argument and answer the substance of it.

Applicants, however, in an effort to advance prosecution, reply to the Office Action with substantially same arguments made in response to the Final Office Action mailed February 23, 2007 ("Final Office Action"), as requested by the Examiner during the telephonic interview.

II. Regarding the Office Action

In the Office Action, the Examiner requested for documents relevant to the claimed invention on SAP Web Dynpro and SAP Dynpro; rejected claims 1, 3-15, 17-18, and 20 under 35 U.S.C. § 102(b) as being unpatentable over SNAP ("Using the SNAP Development Environment," by Template Software) and WEB ("Using the Web Component," by Template Software); and rejected claims 2, 16, and 19 under

35 U.S.C. § 102(b) or 35 U.S.C. § 103(a) as being unpatentable over the commercial product line by Template Software in view of Development Tools.

Applicants have amended claims 2, 16, and 19. Claims 1-20 are currently pending. Based on the foregoing amendments and the following remarks, Applicants respectfully traverse the rejections of the pending claims.

A. Request for Documents

Applicants have amended claims 2, 16, and 19 to remove the terms "SAP Dynpro" and "SAP Web Dynpro." Thus, documents relevant to the claimed invention on SAP Dynpro and SAP Web Dynpro are not required.

B. Rejections of Claims 1, 3-15, 17-18, and 20 Under 35 U.S.C. § 102(b)

The Examiner asserted that the SNAP programming language and the Web Component are "a single offering and constitute a single reference" because "[o]nce Web functionality is present and the Web component is installed the two are inseparable." Final Office Action at 10. The Examiner also asserted that "[s]ince . . . these products work together[,] they constitute a single reference." Office Action at 4. Applicants respectfully submit that the Examiner's assertion is not supported by the references. First, nowhere does SNAP disclose anything related to the Web component, and thus a SNAP application or SNAP Development Environment is operable with or without the Web Component. Further, WEB discloses that the Web Component provides a "gateway, which manages the connection between end user web browsers and the requested, running SNAP application." WEB at 2-4. As shown in detail in Figure 2-1 of WEB, the gateway of the Web Component run **as a separate entity** between a HTTP server and a running SNAP application, or **as a plug-in**

attached to a HTTP server. WEB at 2-4. Thus, the Examiner's assertion that the Web Component and SNAP application are "inseparable" is based on the Examiner's opinion and not supported by any disclosure made by the references. Accordingly, Applicants respectfully request that the Examiner support the assertion based on the references before shifting the burden to Applicants to provide compelling case law that deals with dependencies of layered products.

Moreover, even if SNAP and WEB can be combined to constitute a single reference as asserted by the Examiner, the combination of SNAP and WEB fails to teach or suggest every claim element recited in independent claim 1. In particular, the combination of SNAP and WEB fails to teach or suggest "generating a **converted design-time representation of the application** based on the original design-time representation, the converted design-time representation for use in a second run-time environment for executing applications having been developed in a second design-time environment, the second design-time environment using a second programming model comprising one or more second model elements including models, views, and controllers, the converted design-time representation including one or more application views based on the one or more application screens, and **converted processing logic based on the original processing logic**, the converted processing logic capable of being executed in the second run-time environment." (emphasis added).

WEB discloses that the Web Component "converts . . . SNAP displays **at run time** to HTML or Java displays." WEB at 2-9. (emphasis added). Thus, the Web Component does not generate **a converted design-time representation** of SNAP displays. The Examiner asserted that "the format is not a binary 0's and 1's but HTML

with embedded JAVA . . . stored in [files] . . . can be directly manipulated. Final Office Action at 11. Applicants respectfully disagree.

First, WEB teaches against storing a converted run-time representation of SNAP displays for later manipulation by disclosing that "(t)he SNAP display web pages **exist only as long as** they are accessed by end users." WEB at 2-4. (emphasis added). Thus, the converted run-time representation of SNAP displays exists only at run-time and is not stored as files for manipulation at design-time. Moreover, even if a converted run-time HTML representation of SNAP displays can be stored as files and manipulated later at design-time, a converted **run-time JAVA representation** of SNAP displays is binary (0's and 1's) and thus may not be manipulated later **at design time**. WEB fails to disclose that the Web Component enables a developer to manipulate a converted run-time JAVA representation of SNAP displays. Thus, the Web Component generates, at most, a converted run-time representation of SNAP displays, and fails to generate a converted design-time representation of SNAP displays.

Furthermore, the Web Component generates neither **a converted design time representation** nor **a converted run-time representation** of processing logic of a SNAP application. WEB discloses that converting SNAP application to run on the web does not involve "generating a converted **design-time representation** of the application based on the original design-time representation, . . . the converted design-time representation including . . . **converted processing logic** based on the original processing logic . . . " (emphasis added). Instead, the Web Component generates a converted **run-time representation** of **SNAP displays only** and creates a new object of the WEB ACCEPTER class, which allows a running SNAP application to accept

connections from a HTTP server via the Web Component gateways. Id. Thus, processing logic of a SNAP application runs "**unconverted**" and is only exposed to the web through the converted run-time representation of SNAP displays and the WEB ACCEPTER object. WEB at 2-10. In another words, rather than generating a converted design-time representation of processing logic of a SNAP application, the Web Component merely facilitates exposing the processing logic to the web.

This argument is further supported by WEB at 2-10 and 2-11, which details the steps involved in running a SNAP application on the web. At steps 3 and 4, the Web Component connects to a running SNAP application and "**converts the end-user request into an event or into data that the running SNAP application understands**. . . . The Web Component classes do this conversion . . . [and] **the conversion is transparent to the application's SNAP code**, as well." WEB at 2-10. Thus, rather than converting processing logic of a SNAP application, the Web Component converts end user requests for a SNAP application. Also, because the conversion of the end-user request is transparent to the application's SNAP code, the code does not need to know whether the request is originally from the web or from a standard SNAP client application.

At step 5, "[t]he **SNAP application processes the event or data**, and produces output." WEB at 2-11. (emphasis added). The "SNAP application executes with **little or no knowledge of its connection to the web** . . . [and] executes as though its end users were communicating with the application process directly [and not using the web]." Id. "While the **application generates displays in the same way as a regular SNAP application does**, you [i.e., developer] have developed the **displays for the**

application with the web-imposed features and constraints in mind." Thus, again, there are no changes made to processing logic of a SNAP application to run the application on the web. The processing logic remains "unconverted" within a SNAP application whether the application runs on the web or not, and processes the converted end user request without knowing that the request is from the web. The Web Component further puts the responsibility **on developers** to develop "web-aware" SNAP displays to enable a SNAP application to run on the web.

At step 6, "[t]he Web Component **converts the SNAP displays** generated by the [SNAP] application at run time into HTML and sends them to the Web Component gateway." WEB at 2-11.

For these additional reasons, converting a SNAP application to run on the web does not involve "generating a converted design-time representation of the application based on the original design-time representation, . . . the converted design-time representation including . . . converted processing logic based on the original processing logic . . . "

The Examiner asserted that WEB at 2-3 to 2-5 discloses "converted processing logic." However, WEB at 2-3 to 2-5 discusses generating a run-time representation of SNAP displays only.

> The Web Component enables you to develop SNAP applications whose **displays** are presented to end users as **web pages**. Through the Web Component software, **the SNAP application's displays** are **converted at run time** into **web pages** and **displayed via end users' web browsers**. When end users **access these web pages** through their browsers, they actually **interact with the SNAP application**. For SNAP applications, end users submit URL requests to connect to those applications in the same way that they request access to statically defined web pages; however, the URLs used to connect to SNAP applications specify the applications to be accessed and other necessary related information.

> The difference between simple, statically defined web pages and SNAP application displays presented as web pages is that **SNAP display web pages are dynamically generated** by an interactive program, ... The **SNAP display web pages exist only as long as they are accessed by end users**, while static web pages exist for some significant amount of time between manual updates.
>
> . . . In contrast, **displays used in SNAP applications** that run on the web are **HTML displays or Java applets** generated by the Web Component
>
> **Characteristics of the displays and applets** that run on the web are as follows:
>
> - **At run time**, the **Web Component converts** most kind of **SNAP application displays** to interactive **HTML documents**, and **converts other kinds of SNAP displays** (charts, canvases, and topologies) **to static Java applets**. . . .
> - SNAP applications on the web use sophisticated URL addressing schemes to prevent end users from accessing unauthorized portions of SNAP applications or other end users' application data.
> - **Certain features of SNAP do not operate over the web, because of the structure and limitations of HTML and of the various web browser programs**.

WEB at 2-3.

WEB at 2-4 discusses the structure of SNAP applications on the web: the end users' web browsers, a web server, the Web Component gateway, and a running SNAP application. WEB at 2-5 discusses the end user's web browsers, the web server, and the Web Component gateway in more detail. Therefore, nowhere does WEB at 2-3 to 2-5 teach or suggest "generating a converted design-time representation of the application based on the original design-time representation, . . . the converted design-time representation including . . . converted processing logic based on the original processing logic . . . "

For at least these reasons, SNAP and WEB, taken alone or in combination, fail to teach or suggest every claim element recited in independent claim 1. Independent

claims 10, 15, and 18 recite features that are similar to the features recited in independent claim 1. Accordingly, Applicants respectfully request reconsideration and withdrawal of the § 102 rejection of independent claims 1, 10, 15, and 18 based on SNAP and WEB.

Claims 3-9 depend from claim 1, claims 11-14 depend from claim 10, claim 17 depends from claim 15, and claim 20 depends from claim 18. Thus, dependent claims 3-9, 11-14, 17, and 20 are allowable by virtue of their dependence on an allowable independent claim. Accordingly, Applicants respectfully request reconsideration and withdrawal of the § 102 rejection of dependent claims 3-9, 11-14, 17, and 20 based on SNAP and WEB.

C. Rejections of Claims 2, 16, and 19 Under 35 U.S.C. §§ 102(b) or 103(a)

The Examiner rejected claims 2, 16, and 19 under 35 U.S.C. § 102(b) or 103(a) "as being unpatentable over the commercial product line by Template Software in view of Development Tools." Final Office Action at 8. Claim 2 depends from claim 1, claim 16 depends from claim 15, and claim 19 depends from claim 18. Accordingly, claims 2, 16, and 19 are allowable at least by virtue of their dependence on allowable independent claims 1, 15, and 18, respectively.

With respect to the § 103 rejection, the Examiner asserted that "Template teaches the ability to build GUIs in **SNAP** to run as a local application and how to Web enable them with the product **WEB** which enables them to run in another environment." Final Office Action at 8. Applicants respectfully submit that building a local application and "Web enabling" the local application is not the same as and does not necessarily involve "generating a converted **design-time representation of the application** based

on the original design-time representation . . . " (emphasis added). As explained above with respect to independent claim 1, WEB teaches web enabling an application by converting an end-user request to an event or data that the application understands and converting output displays generated by the application at run-time to HTML pages without generating a converted design-time representation of the application. Therefore, the Examiner has not established *prima facie* case of obviousness with respect to claim 2, 16, and 19. Accordingly, Applicants respectfully request reconsideration and withdrawal of the §§ 102 or 103 rejection of claims 2, 16, and 19.

III. Conclusion

In view of the foregoing remarks, Applicants respectfully request reconsideration and reexamination of this application and the timely allowance of the pending claims.

Please grant any extensions of time required to enter this response and charge any additional required fees to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: November 1, 2007

By: _____

Jeffrey A. Berkowitz
Reg. No. 36,743
(202) 408-4000